Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 82 (2016) 99 – 106

Procedia Computer Science

Symposium on Data Mining Applications, SDMA2016, 30 March 2016, Riyadh, Saudi Arabia

Performance Comparison of Spark Clusters Configured Conventionally and a Cloud Service

Hameeza Ahmed[a], Muhammad Ali Ismail[a,*], Muhammad Faraz Hyder[a], Syed Muhammad Sheraz[a], Nida Fouq[a]

[a]*High Performance Computing Centre (HPCC), Department of Computer & Information Systems Engineering, NED University of Engineering & Technology University Road, Karachi-75270, Pakistan*

Abstract

Apache Spark is an open source cluster computing technology specifically designed for large scale data processing. This paper deals with the deployment of Spark cluster as a cloud service on the OpenStack based cloud. HiBench benchmark suite is used to compare the performance of Spark cluster as a service and conventional Spark cluster. The results clearly depict how Spark as a cloud service gives more promising outcomes in terms of time, effort and throughput.

Peer-review under responsibility of the Organizing Committee of SDMA2016

Keywords: OpenStack; Spark Cluster; HiBench; Cloud Service; Benchmark;

1. Introduction

With the recent developments in technology, there are numerous sources which are contributing towards the generation of huge amount of data. These sources include now sensor equipped areas, RFID, social networks, large-scale eCommerce, phones, credit cards, atmospheric science, medical records, trains, buses, biological, astronomy, genomics, military surveillance, video archives, photography archives, and the internet of things. Big data means data that's *too fast*, *too big*, or *too hard* for existing tools to process in fact it refers to datasets whose size is beyond the ability of classical database software tools to manage, capture, store, and analyze. Big data is not a single technology, but it involves many of existing fundamental concepts such as parallel processing, distributed file systems, in-memory databases virtualization, and many more [1], [8].

Big data computing is a big challenge in the present era in various aspects including storage, processing, management, analytics, visualization etc. Amongst this data processing is the most challenging aspect. In order to

Peer-review under responsibility of the Organizing Committee of SDMA2016
doi:10.1016/j.procs.2016.04.014

process such huge quantity of data, there exists a variety of programming frameworks namely Apache Hadoop, Apache Spark, HPCC, HPCC Systems (High Performance Computing Cluster), Storm, Lambdoop etc [2].

Apache Spark is an open source big data programming model. It was started as a research project in the AMPLab at UC Berkeley. It is a general-purpose cluster computing engine with libraries for streaming, machine learning, and graph processing. Additionally, it has APIs in Java, Python, and Scala. Spark is an open source big data framework primarily designed for three major objectives namely ease of use, sophisticated analytics, and speed. The processing speed of Spark is increased due to its feature of in-memory cluster computing [3], [4], [5].

With the complexities and challenges involved in big data computing, the need for large computational infrastructure, expensive software, and effort is also raised. Cloud computing provides the ultimate solution to these problems. It does so by providing on-demand resources and charging as per the actual resource consumption. Besides, it allows the infrastructures to be scaled down and up rapidly, adjusting the system to the actual demand [6]. Cloud computing is powerful enough to perform complex and massive-scale computing. It eliminates the need of onsite maintenance of expensive dedicated space, computing hardware, and software [7].

The primary objective of this paper is the deployment of Apache Spark cluster as a cloud service (SAAS) on OpenStack cloud. There are several benefits of providing Apache Spark as SAAS namely scalability, backup and restore facility, ease of use, high speed, increased throughput, lower cost and many others [8]. The work being presented in this paper makes an in-depth analysis of the performance of Spark cluster as a SAAS. It does so by comparing the results of a Spark cluster configured as cloud service with the conventional one. The comparison is done using a complete big data benchmark suite known as HiBench. In order to perform the comparison, total nine benchmarks are executed on both the implementations of Spark cluster. These benchmarks include four separate categories namely Micro benchmarks, Web Search, Machine Learning and Analytical Querying. The benchmarks namely Sort, WordCount, Sleep, and TeraSort are included in micro benchmarks category. The Web Search benchmarks include PageRank while the Machine Learning category is comprised of Bayesian Classification. The analytical query involves three benchmarks namely Hive Join, Scan and Hive Aggregate respectively [9]. The final results clearly depict how apache Spark cluster deployed on OpenStack dominates the conventional cluster both in terms of speed and throughput.

Rest of the paper is organized as follows: Section 2 discusses related work. Apache Spark cluster deployment both as a SAAS on cloud and the conventional cluster is shown in section 3. Section 4 provides description about the benchmark suite HiBench and performance metrics used in the experiment. A detailed comparison of Spark performance results is covered in Section 5. The final conclusion is presented in section 6.

2. Related Work

OpenStack and Apache Hadoop represent the largest open source communities. Their integration will benefit users of both of these communities. Considering this, there has been some efforts in this area most notable of them is the project SAHARA [11]. This integration manages and configures Hadoop cluster in the cloud. The project SAHARA has now been extended for Spark Support but currently Spark can only installed in standalone mode, with no YARN or Mesos support [12]. Our research is novel in the way that this research is succeeded in implementing Spark cluster as a service in an Openstack cloud in distributed mode with full YARN support.

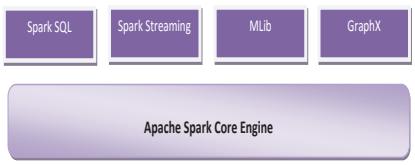


Fig. 1. Components of Spark [4]

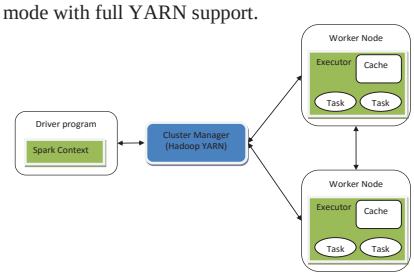


Fig. 2. Apache Spark Cluster Manager [4]

3. Apache Spark Cluster Deployment

Apache Spark is an open source, expressive, fast, and general-purpose cluster-computing project that was created at AMPLabs in UC Berkeley in 2009. It has emerged as a top level Apache project. It is a powerful processing tool providing sophisticated analytics, ease of use, and high speed. It provides high-level APIs in Scala, Java, R and Python. Also, an optimized engine is given in Spark to support general execution graphs. Additionally, there is support for a rich set of higher-level tools namely MLlib for machine learning, GraphX for graph processing, Spark Streaming, and Spark SQL for SQL and structured data processing. A wide range of workloads is covered by Spark including interactive queries, batch applications, iterative algorithms, and streaming. Also, by using Spark framework the management burden of maintaining separate tools is reduced [3], [4], [5]. The current Apache Spark stack supports multiple components as shown in Fig 1.

In order to conduct the experiment, two different spark clusters with same structure were configured, one as a cloud service and other on a conventional machines. Spark clusters are configured on Hadoop as it uses Hadoop's Distributed File System (HDFS) for data storage and Hadoop's YARN for cluster management respectively as shown in Fig 2. Apache Spark cluster requires both the Hadoop and Spark daemons to get started. On master node four daemon processes namely namenode, secondarynamenode, resourcemanager, and master are started. While on slave node three daemon processes including nodemanager, datanode, and worker are started. The configuration details of Spark cluster are listed in Table 1.

Spark cluster configured for cloud service, was setup on an OpenStack based cloud. The cloud is rich in the resources as it is comprised of one controller node, one network node and fifteen compute nodes. for the underlying hardware, each node has a Quad-Core AMD Opteron™ processor with 8 GB RAM and around 200 GB disk along with Ubuntu server 14 serving as the operating system. The controller node configuration is twice of this one. The four separate instances are spawned in order to configure a 4-node Spark cluster as a service, where one node acts as the master and the other three nodes act as slaves. The individual nodes on the cloud based Spark cluster has 4 GB RAM, 50GB Disk , 2 cores and Ubuntu server 14 serving as the operating system.

Similarly, base machines were setup in order to configure the conventional Spark cluster. The underlying hardware of the nodes is based on AMD Opteron™ processor, 4 GB RAM, 20 GB Disk, and 2 cores along with Linux CentOS Release 6.5 64 bit Kernel serving as the operating system. The 4-node Spark cluster is configured where one node acts as the master and the other three nodes act as the slaves.

4. Spark Benchmarks and Performance Metrics

This section mainly discusses a benchmark suite known as HiBench which is used to test the performance of Spark cluster. Also, the performance metrics namely elapsed time, throughput, and speedup are discussed here.

4.1. HiBench Benchmark Suite

HiBench is a comprehensive and representative benchmark suite for Hadoop, Spark, Storm, Storm-Trident and Samza. It consists of a set of programs including both real-world applications and synthetic micro-benchmarks. For each workload, the input data of benchmarks is automatically generated by using prepare scripts. Presently, HiBench contains thirteen workloads which are classified into five categories [9], [10].

HiBench provides four different workloads for Micro benchmarks in Spark. All four are used in the experiment being discussed in the paper. The description about these benchmarks is as follows:

- Sort: It Sorts the *text* input data, which is produced using RandomTextWriter.
- WordCount: It counts the existence of each word in the input data, which are produced using RandomTextWriter.
- TeraSort: TeraSort is a standard benchmark created by Jim Gray. Its input data is generated by Hadoop TeraGen.
- Sleep: This workload tests the framework scheduler by sleeping an amount of seconds in each task.

HiBench provides two workloads namely NutchIndexing, PageRank for Web Search benchmarks. The PageRank used in this experiment as it is responsible to benchmark PageRank algorithm implemented in Spark-

MLLib/Hadoop. The data source was generated from Web data. In case of Machine Learning, HiBench provide following two workloads namely Bayesian Classification and K-means Clustering. The Bayesian classification was used in this paper; it is responsible to benchmark a popular Classification algorithm known as NaiveBayesian. For the task of analytical query three benchmarks namely Join, Scan and Aggregate are available in HiBench. They are responsible to perform the typical OLAP queries by Hive queries. Also, it requires automatically generated web data as input source. All of these three benchmarks have been used in this paper.

Hence, total nine benchmarks of HiBench suite are used in this paper including WordCount, TeraSort, Sleep, Sort, Bayesian, Aggregate, Join, PageRank, and Scan respectively. The input data is varying as it is generated automatically for each benchmark using the prepare script. The computations have been performed multiple times, thus the average result values are reported in this paper for both the Spark clusters.

4.2. Performance Metrics

- Elapsed Time: It is the time required to perform an event. It is the difference between beginning time and an ending time. It can differentiate the performance as less elapsed time indicates good performance. It is measured in seconds.
- Throughput: It is the amount of work that can be performed in a given period of time. It is measured in bytes/seconds.
- Speedup: It is the ratio of T_1 over T_N which is elapsed time of *1* and *N* workers.

5. Result Analysis

The work being presented in this paper mainly deals with the implementation of Apache Spark cluster as SAAS on OpenStack cloud. In order to study the effectiveness of Spark as SAAS a comparison is done between Spark cluster on cloud and conventional Spark cluster using a HiBench benchmark suite.

Table 2 shows the final statistics namely input data size, duration, throughput for each specific benchmark when executed on a conventional cluster. Table 3 shows the final outcomes for each specific benchmark when executed on Spark cluster configured as a cloud service. When the benchmark Aggregate is executed on cloud, it shows more promising outcome both in terms of duration and throughput as compared to conventional cluster. It can be clearly seen through Fig 3 how the elapsed time of Aggregate when executed on cloud as SAAS is much lower as compared to the conventional Spark cluster. Similarly, the TeraSort algorithm shows major improvement when executed on cloud, as the elapsed time is reduced by a significant portion. Similar results can also be observed for benchmarks like Join, PageRank, Scan, Sort, and WordCount. Again in all of these benchmarks execution, Spark on cloud gives less elapsed time as compared to conventional cluster scenario. The cloud based Spark cluster gives less elapsed time for benchmark Bayesian as compared to conventional cluster. Finally, in case of benchmark Sleep again Spark as a SAAS leads its counterpart. Although, there is no I/O involved in benchmark Sleep but it consumes significant proportion of CPU time in its busy wait state. In terms of throughput involving both throughput in terms of bytes/sec and throughput/node, all the benchmarks executed on OpenStack cloud as a SAAS shows higher throughput as compared to Spark conventional cluster. Fig 4. Shows bars depicting throughput in bytes/sec whereas Fig 5 represents a bar chart showing throughput/node. It can be clearly observed through Fig. 4 and 5 how Spark as a SAAS is much promising having the larger bars as compared to Spark conventional cluster. The speedup is also calculated for both the scenarios which clearly depict the dominance of Spark as a cloud service as compared to conventional cluster. The benchmark TeraSort leads the other benchmarks in terms of speedup which is 3.384.

The performance of both the clusters can further be analysed by Table 4 which shows the summarized CPU usage and system load for all the nine benchmarks discussed earlier. In case of Micro benchmarks, CPU is processing mostly system or user tasks with the little involvement of I/O wait as well. But, the benchmark TeraSort is showing much higher involvement of processor in the I/O wait. It is clear from Table 4 that Micro workloads are highly CPU bound and slightly I/O bound with the exception of TeraSort which shows much greater I/O activity as compared to others. The benchmark Sleep does idle wait; as it can be seen the CPU is showing activity for a finite duration only. The PageRank benchmark involves more CPU activity for executing the user tasks. It also involves I/O wait but it is less. Similarly, for Bayesian workload the user task dominates the CPU processing time. The

workloads in analytical query are also CPU bound, with the processor being actively involved in user and system tasks. Also, the I/O wait is minimal in analytical query. On the whole, the Spark cluster as a cloud service shows less CPU utilization as compared to conventional Spark cluster. Also, conventional Spark cluster is more burdened in terms of system load as compared to Spark as a service.

Table 1. Spark Cluster Benchmark HiBench Run Time Parameters

Properties	**Values**
Spark Version	Spark 1.5.2 built for Hadoop 2.4.0
No of Nodes	4 (1 Master and 3 Slaves/Workers)
Hadoop Version	Hadoop-2.4.0
HiBench Version	HiBench-5.0
Benchmarks	WordCount, TeraSort, Sleep, Sort, Bayesian, Aggregate, Join, PageRank, Scan
Hadoop Replication Factor	2
Number of Executor	2
Executor Memory	2 GB
Driver Memory	1 GB

Table 2. Spark HiBench Execution Results on Conventional Cluster

Benchmarks	**Input Data Size(bytes)**	**Duration(s)**	**Throughput(bytes/s)**	**Throughput/node**
Aggregate	37276833	128.082	291038	97012
Bayesian	450822628	231.789	1944969	648323
Join	192861180	145.934	1321564	440521
PageRank	259928115	576.603	450792	150264
Scan	183579314	122.945	1493182	497727
Sleep	0	329.042	0	0
Sort	328490396	88.918	3694307	1231435
TeraSort	3200000000	461.694	6930997	2310332
WordCount	2204456452	142.487	15471281	5157093

Table 3. Spark HiBench Execution Results on Cloud

Benchmarks	**Input Data Size(bytes)**	**Duration(s)**	**Throughput(bytes/s)**	**Throughput/node**	**Speedup**
Aggregate	37276833	90.990	409680	136560	1.4076
Bayesian	375706036	99.165	3788695	1262898	2.3374
Join	194078124	101.190	1917957	639319	1.4421
PageRank	259928115	346.854	749387	249795	1.6623
Scan	184796438	93.425	1978019	659339	1.3159
Sleep	0	301.696	0	0	1.0906
Sort	328490787	53.018	6195835	2065278	1.6771
TeraSort	3200000000	136.431	23455079	7818359	3.3840
WordCount	2204457594	68.753	32063438	10687812	2.0724

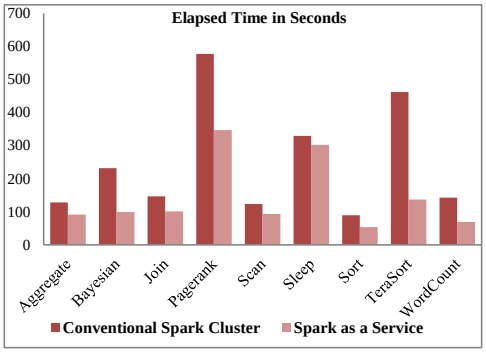


Fig. 3. Spark Cluster Benchmarks Execution Time

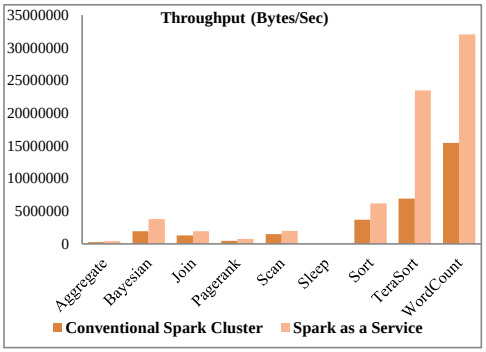


Fig. 4. Spark Cluster Benchmarks Throughput

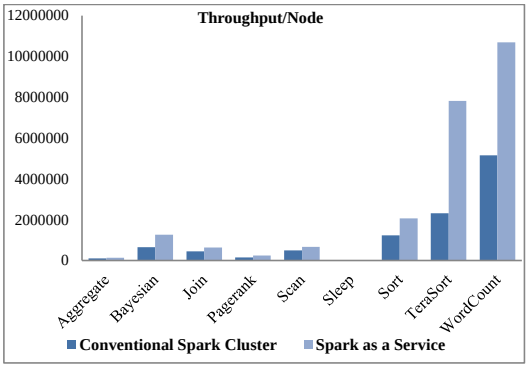
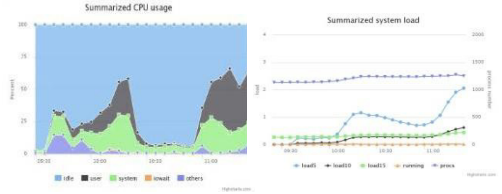
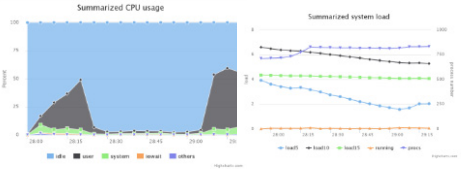
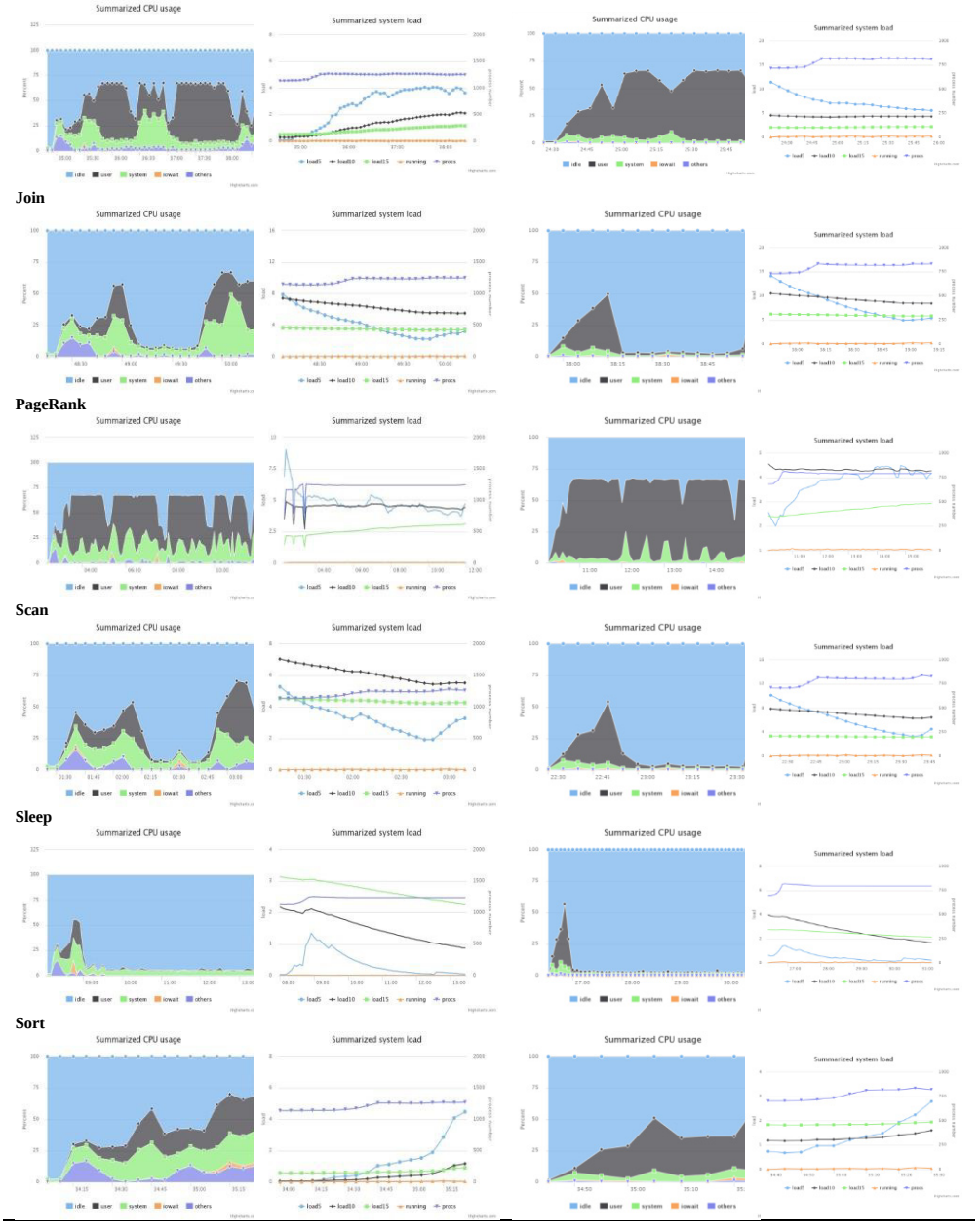
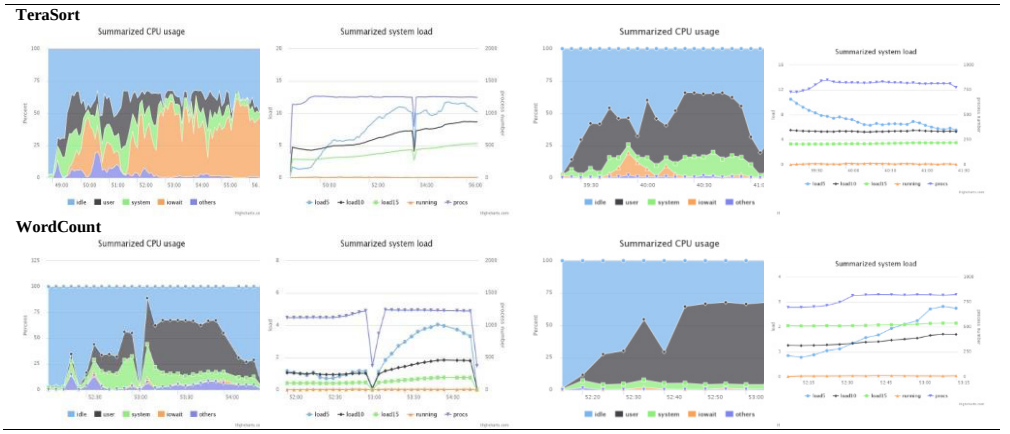


Fig. 5. Spark Cluster Benchmarks Throughput/Node

Table 4. Summarized CPU usage and system load for various HiBench Benchmarks

	Spark Conventional Cluster	Spark Cluster as a Service
Aggregate		
Bayesian		


Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load
Join
Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load
PageRank
Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load
Scan
Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load
Sleep
Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load
Sort
Summarized CPU usage
Summarized system load
Summarized CPU usage
Summarized system load



6. Conclusions

The paper clearly highlights the impact of running Spark cluster as a SAAS. The effectiveness of Spark cluster as a SAAS is observed by comparing the outcomes of a HiBench benchmark suite on both the cloud and conventional cluster. The final results clearly show Spark as a SAAS is more effective both in terms of speed and throughput. Also, the Spark as a SAAS shows less CPU utilization and system load when executed with HiBench benchmark suite involving the four major workload categories namely Micro benchmarks, Analytical Query, Web Search, and Machine Learning respectively.

Acknowledgements

This research work is supported by the High Performance Computing Centre (HPCC), NED University of Engineering and Technology, Pakistan.

References

1. H. Ahmed, M. A. Ismail, and M. F. Hyder, “Performance Optimization of Hadoop cluster using Linux Services,” in *Proceedings of the 17th IEEE International Multi-Topic Conference (INMIC)*, Karachi, Pakistan,8-10 December, 2014, pp. 167–172.
2. C. Dobre and F. Xhafa, “Parallel Programming Paradigms and Frameworks in Big Data Era,” *International Journal of Parallel Programming*, vol. 42, no. 5, 2014, pp. 710–738.
3. S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, “A survey of open source tools for machine learning with big data in the Hadoop ecosystem,” *Journal of Big Data*, vol. 2, no. 1, 2015, pp. 1–36.
4. “Apache Spark.” [Online]. Available: http://Spark.apache.org/ [January. 1, 2016].
5. S. Gopalani and R. Arora, “Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means,” *International Journal of Computer Applications*, vol. 113, no. 1, 2015.
6. Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A. and Buyya, R., 2015. Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, *79*, pp.3-15.
7. Hashem, Ibrahim Abaker Targio, et al. "The rise of “big data” on cloud computing: review and open research issues." *Information Systems* 47 (2015): 98-115.
8. M. F. Hyder, M. A. Ismail, and H. Ahmed. "Performance comparison of Hadoop Clusters configured on virtual machines and as a cloud service," in *Proceedings of the 10th IEEE International Conference on Emerging Technologies (ICET)*, Islamabad, Pakistan, 8-10 December, 2014, pp. 60-64.
9. “HiBench Benchmark Suit.” [Online]. Available: https://github.com/intel-hadoop/HiBench [January. 1, 2016].
10. Huang, Shengsheng, et al. "Hibench: A representative and comprehensive hadoop benchmark suite." *Proc. ICDE Workshops*. 2010.
11. “Sahara Project.” [Online]. Available: http://docs.openstack.org/developer/sahara/ [March. 3, 2016].
12. “Spark Plugin for Sahara.” [Online]. Available: http://docs.openstack.org/developer/sahara/userdoc/spark_plugin.html [March. 3, 2016].